The SPEKE Protocol Revisited

Feng Hao, Siamak F. Shahandashti

Newcastle University, UK
{feng.hao, siamak.shahandashti}@ncl.ac.uk

Abstract. The SPEKE protocol is commonly considered one of the classic Password Authenticated Key Exchange (PAKE) schemes. It has been included in international standards (particularly, ISO/IEC 11770-4 and IEEE 1363.2) and deployed in commercial products (e.g., Blackberry). We observe that the original SPEKE specification is subtly different from those defined in the ISO/IEC 11770-4 and IEEE 1363.2 standards. We show that those differences have critical security implications by presenting two new attacks on SPEKE: an impersonation attack and a key-malleability attack. The first attack allows an attacker to impersonate a user without knowing the password by engaging in two parallel sessions with the victim. The second attack allows an attacker to manipulate the session key established between two honest users without being detected. Both attacks are applicable to the original SPEKE scheme, and are only partially addressed in the ISO/IEC 11770-4 and IEEE 1363.2 standards. We highlight deficiencies in both standards and suggest concrete changes.

1 Introduction

Password Authenticated Key Exchange (PAKE) is a protocol that aims to establish a secure communication channel between two remote parties based on a shared low-entropy password without relying on any external trusted parties. Since the seminal work by Belloven and Merrit in 1992 [2], many PAKE protocols have been proposed, and some have been standardised [10, 11].

The Simple Password Exponential Key Exchange (SPEKE) protocol is one of the most well-known PAKE solutions. It was originally designed by Jablon in 1996 [7]. Although some concerns have been raised [8, 9], no major flaws seems to have been uncovered. Over the past decade, SPEKE has been included in the IEEE P1362.2 [10] standard draft[1] and ISO/IEC 11770-4 [11]. Furthermore, SPEKE has been deployed in commercial applications – for example in BlackBerry devices produced by Research in Motion [6] and in Entrust's TruePass end-to-end web products [5].

In this paper, we revisit the original SPEKE protocol and review its specifications in the two standardisation documents: IEEE P1363.2 and ISO/IEC 11770-4. We observe that the original protocol is subtly different from those

[1] At the time of writing, the latest draft available on the IEEE P1363.2 website is D26. See `http://grouper.ieee.org/groups/1363/passwdPK/draft.html`

defined in the standards. The reason for the difference, or deviation from the original specification, is not justified clearly in the standards.

During this investigation, we have identified several issues with SPEKE that have not been reported before. Our findings are summarised below:

1. We show that the original SPEKE protocol is subject to an impersonation attack when the victim is engaged in two parallel sessions with an active attacker. The attacker is able to achieve mutual authentication in both sessions without knowing the password.
2. We show that the original SPEKE protocol is subject to a key-malleability attack. The attacker, sitting in between two honest users, is able to manipulate the session key without being detected.
3. While both attacks clearly succeed against the original SPEKE protocol, we show they are partially addressed in IEEE P1363.2 and ISO/IEC 11770-4, but not in any rigorous manner. We propose explicit and concrete changes to both standards.

Details of our findings are explained in the following sections.

2 The original SPEKE scheme

First, we define the original SPEKE scheme based on Jablon's 1996 paper [7]. Let p be a safe prime, $p = 2q+1$ where q is also a prime. Assume two remote parties, Alice and Bob, share a common password s. SPEKE defines a function $f(\cdot)$ to map a password s to a group element: $f(s) = s^2 \bmod p$. We use g to denote the result returned from $f(s)$, i.e., $g = f(s)$. The SPEKE protocol provides implicit authentication in one round, which is defined below. (Unless stated otherwise, all modular operations are performed modulo p, hence the explicit $\bmod p$ is omitted for simplicity.)

SPEKE (one round) *Alice selects* $x \in_R [1, q-1]$ *and sends* g^x *to Bob. Similarly, Bob selects* $y \in_R [1, q-1]$ *and sends* g^y *to Alice.*

Upon receiving the sent data, Alice verifies that g^y is within $[2, p-2]$. This is to ensure the received element does not fall into the small subgroup of order two, which contains $\{1, p-1\}$. Alice then computes a session key $\kappa = H((g^y)^x) = H(g^{xy})$, where H is a secure one-way hash function. Similarly, Bob verifies that g^x is within $[2, p-2]$. He then computes the same session key $\kappa = H((g^x)^y) = H(g^{xy})$.

To provide explicit key confirmation, the SPEKE paper defines the following procedure. One party sends $H(H(\kappa))$ and the other party replies with $H(\kappa)$. The paper does not specify who must initiate the key confirmation and hence leaves it as a free choice for specific applications to decide.

3 Previously reported attacks

In 2004, eight years after SPEKE was initially designed, Zhang presented an exponential-equivalence attack [8]. The attack is based on the observation that some passwords are exponentially equivalent. Hence, an active attacker can exploit that equivalence to test multiple passwords in one protocol execution. This is especially problematic when the password is digits-only, e.g., a Personal Identification Numbers (PIN). As a countermeasure, Zhang proposed to hash the password before taking the square operation. In other words, he redefined the password mapping function as: $f(s) = (H(s))^2 \bmod p$. The hashing of passwords makes it much harder for the attacker to find exponential equivalence among the hashed outputs. Zhang's attack is acknowledged in IEEE P1363.2 [10], which adds a hash function in SPEKE when deriving the base generator from the password.

In 2005, Tang and Mitchell presented three attacks on SPEKE [9]. The first attack is similar to Zhang's [8] – an on-line attacker tests multiple passwords in one execution of the protocol by exploiting the exponential equivalence of some passwords. The second attack assumes that the user shares the same password with two servers, say $S1$ and $S2$. By relaying the messages between the client and $S2$, the attacker may trick the client into believing that she shares a key with $S1$, but actually the key is shared with $S2$. The authors call this an "unknown key-share" attack. They suggest to address this attack by including the server's identifier into the computation of g. (However, we note that this suggested countermeasure has the side-effect of breaking the symmetry of the original protocol.) The third attack indicates a generic vulnerability. In this scenario, two honest parties launch two concurrent sessions. The attacker can swap the messages between the two sessions to exchange the two session keys. The two communicating parties will be able to decrypt messages successfully but they may get confused about which message belongs to which session.

4 New attacks

In this section, we describe two new attacks: an impersonation attack and a key-malleability attack. The first attack indicates a practical weakness in the original design of SPEKE, while the second attack has an unfavourable implication on the theoretical analysis of the protocol.

4.1 Impersonation attack

The impersonation attack works when the user is engaged in several sessions in parallel with another user. This is a realistic scenario in practice as two users may want to run several concurrent SPEKE key exchange sessions and use each established channel for a specific application, as explained by Tang and Mitchell [9].

We assume Alice and Bob share a common password. Their respective identities are denoted by $\hat{A}$ and $\hat{B}$. Without loss of generality, we assume Alice

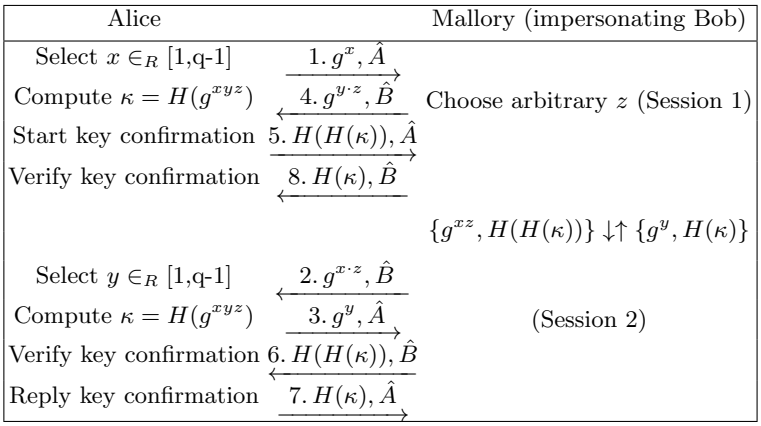

Fig. 1. Impersonation attack on SPEKE

initiates a SPEKE session – which we call Session 1 – with Bob by sending g^x (see Figure 1; we append the sender's purposed identity in the key exchange flow to make the illustration of the attack clearer). But the message is intercepted by Mallory. Mallory chooses an arbitrary z from $[2, p-2]$ and raises the intercepted g^x by the power of z to obtain g^{xz}. Pretending to be "Bob", Mallory initiates another SPEKE session – which we call Session 2 – with Alice by sending g^{xz}. The use of z serves to make the messages different between the two sessions. In the second session, Alice replies with g^y. Mallory raises this item to the power of z to obtain g^{yz}, and sends the result to Alice as the reply in Session 1. Following the key confirmation procedure as in the original SPEKE paper, Alice provides the first key confirmation challenge in Session 1 $H(H(\kappa))$, which is subsequently relayed to Session 2 as Bob's key confirmation challenge. In Session 2, Alice answers the key confirmation challenge by replying with $H(\kappa)$, which is then relayed in Session 1 to complete the mutual authentication in both sessions. Recall that in a Password Authenticated Key Exchange protocol, the notion of "authentication" is defined based on the knowledge of a secret password. However, without knowing the password, Mallory has been successfully authenticated by Alice as "Bob", someone who supposedly shares the exclusive knowledge with Alice about a secret password.

In essence, this impersonation attack follows the "wormhole attack" [3], in which the attacker relays the sender's message back to the sender in order to pass authentication. However, the "wormhole attack" presented in [3] works in a PKI-based key exchange setting while the attack reported here occurs in a password-based key exchange setting. The two settings are distinct. Nonetheless, both attacks highlight the importance of including explicit user identities in the authenticated key exchange process.

To some extent, this impersonation attack is similar to the "unknown key-share" attack described in Tang-Mitchell's paper [9]. However, our attack seems to be more feasible and more harmful than theirs. The main difference is that

in our attack, the attacker changes the user's message and sends the modified message back to the user herself (instead of to a third party as in [9]). At the end of the key establishment process, Alice thinks she is sharing a session key with the real "Bob", but she is actually sharing the key with another instance of herself. This confusion of identity in the key establishment can cause problems in some scenarios. For example, using the derived session key κ in an authenticated mode (e.g., AES-GCM), Alice may send an encrypted message to "Bob": "*Please pay Charlie 5 bitcoins*". But Mallory can relay the message back to Alice in the second session. Since the message is verified to be authentic from "Bob", Alice may follow the instruction and pay Charlie instead (in a practical application, Alice is likely an automated program that follows the protocol). Thus, although Alice's initial intention is to make "Bob" pay Charlie 5 bitcoins, she ends up paying Charlie instead. In this attack, the supposed "Bob" seems to be liable but the real Bob is actually never involved.

4.2 Key-malleability attack

A second attack is called the key-malleability attack. In this attack, the attacker sits in the middle between two honest users (see Figure 2). The attacker chooses an arbitrary z within the range of $[2, q-1]$, raises the intercepted item to the power of z and passes it on. The users at two ends are still able to derive the same session key $\kappa = H(g^{xyz})$, but without being aware that the messages have been modified.

We do not claim there is a direct practical harm caused by this attack. However, the fact that an attacker is able to manipulate the session key without being detected may have significant implications on the theoretical analysis of the protocol. In the original SPEKE paper, the protocol comes with no security proofs[2]. However, it is heuristically argued that the security of the session key in SPEKE depends on either the Computational Diffie-Hellman assumption (i.e., an attacker is unable to compute the session key) or the Decisional Diffie-Hellman assumption (i.e., an attacker is unable to distinguish the session key from random). The existence of such a key-malleability attack suggests that a tight reduction to CDH or DDH is impossible. The attacker's ability to inject randomness into the session key without being noticed can significantly complicate the theoretical analysis. As an example, let us assume the attacker chooses

[2] In 2001, MacKenzie published a manuscript "On the Security of the SPEKE Password-Authenticated Key Exchange Protocol" on IACR ePrint 2001/057. Although an ePrint manuscript is usually not regarded as a formal peer-reviewed publication, it is cited in IEEE P1363.2 [10] to support the argument that the SPEKE protocol has been formally proved to be secure based on some number theoretical assumptions. We observe that the formally analysed SPEKE is substantially different from the original SPEKE: e.g., 1) the derivation of the generator is different, and as a result, the protocol is two-round instead of one-round as in the original SPEKE paper or the standards; 2) the definition of the key confirmation function is different from that in the original paper or standards; 3) it regards the key confirmation as mandatory rather than optional.

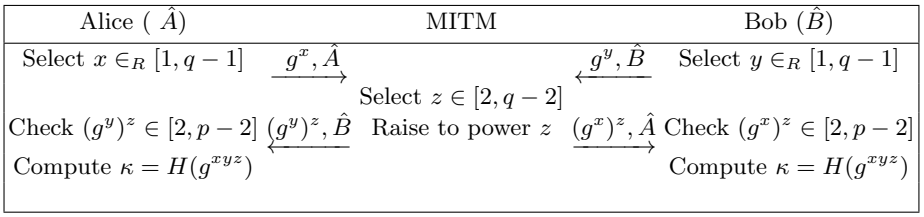

Fig. 2. Key-malleability attack on SPEKE

z as a result of an arbitrary function with the incepted inputs, i.e., $z = f(g^x, g^y)$. Because of the correlation of items on the exponent, standard CDH and DDH are no longer applicable here (recall that in the CDH/DDH assumption, the secret values on the exponent are assumed to be independent).

5 Discussion

While the two attacks clearly work on the original SPEKE protocol [7], it may be arguable whether they are applicable to the variants of SPEKE defined in IEEE P1363.2 and ISO/IEC 11770-4. In this section, we explain the difference between the original protocol and its variants in the standards in relation to the two attacks.

5.1 Explicit key confirmation

First of all, we observe that the key confirmation procedure of SPEKE defined in the standards is different from that in the original SPEKE paper [7]. For example, in ISO/IEC 11770-4, the key confirmation works as follows [11] (the procedure in IEEE P1363.2 [10] is basically the same).

$$\begin{aligned} \text{Alice} \rightarrow \text{ Bob } &: H(\text{“}0x03\text{”} \| g^x \| g^y \| g^{xy} \| g) \\ \text{Bob } \rightarrow \text{Alice} &: H(\text{“}0x04\text{”} \| g^x \| g^y \| g^{xy} \| g) \end{aligned}$$

As explicitly stated in the ISO/IEC 11770-4 standard, there is no order in the above two steps[3]. Either party is free to send out the key confirmation message without waiting for the other party.

Effect on impersonation attack We observe that the above key confirmation procedure does not prevent the impersonation attack. The attacker is still able to relay the key confirmation string in one session to another parallel session to accomplish mutual authentication in both sessions without being detected. The attack works largely because the session keys are identical in the two sessions.

[3] In the same standard, it is also stated that there is no order during the SPEKE exchange phase. We find the two statements contradictory: the fact that g^x comes before g^y in the definition of key confirmation implies there is an order during the key exchange phase.

Effect on Key-malleability attack The key-malleability attack no longer works with the key confirmation procedure defined in ISO/IEC 11770-4 (and IEEE P1363.2). However, it is worth noting that the key confirmation procedures in both standards are marked as "optional". Hence, the key-malleability attack is not completely addressed.

5.2 Definition of Password

In the original SPEKE paper, the mapping of a password s to a group element over the prime field is simply achieved by $f(s) = s^2$. To prevent Zhang's exponential-equivalence attack, it is necessary to add a hash function before performing the squaring operation, i.e., $f(s) = (H(s))^2$. This is essentially the mapping function defined in ISO/IEC 11770-4 and IEEE P1363.2 (for the case that p is a safe prime). However, the definition of the shared secret is subtly changed in both standards. For example, in IEEE P1363.2 [10], the shared low-entropy secret (denoted π in the standard document) is defined as follows:

> *"A password-based octet string, used for authentication. π is generally derived from a password or a hashed password, and may incorporate a salt value, identifiers for one or more parties, and/or other shared data."*

It is worth nothing that in the above definition, the incorporation of "a salt value, identifiers for one or more parties, and/or other shared data" is not mandatory (as indicated by the use of the word "may").

In ISO/IEC 11770-4 [11], the shared low-entropy secret is defined as follows with an additional note:

> *"A password-based octet string which is generally derived from a password or a hashed password, identifiers for one or more entities, an identifier of a communication session if more than one session might execute concurrently, and optionally includes a salt value and/or other data.*
>
> *NOTE - it is required to include one or more the entity identifiers and a unique session identifier into the value of π, in order to avoid that a key establishment mechanism might be vulnerable to an unknown key share attack addressed in [TC05]."*

The above definition seems to include the "identifiers for one or more entities" as part of the shared secret. However, the standard does not provide any formula. It is not even clear if one or both entities' identifiers should be included, and if only one identifier needs to be included, which one and how. Furthermore, the word "generally" weakens the rigour in the definition and makes it subject to potentially different interpretations. The note below the definition states that: the inclusion of the entity/session identifiers is required to address the UKS attack in [TC05] [9]. However, the UKS attack reported in [9] works under the assumption that the user shares the same password with two different servers. In many applications, it is often considered reasonable to exclude that assumption

from the threat model (otherwise, the solution may become overly complex). In those cases, the justification becomes no longer valid. Therefore, on whether the entity/session identifier "should", "must" or "may" be included as part of the shared secret, we find the current ISO/IEC 11770-4 standard not sufficiently clear.

Effect on impersonation attack Strictly speaking, if the entity identifiers (or the session identifiers) are included in the definition of the shared secret, the impersonation attack presented in Section 4 will not work. In the IEEE definition, the inclusion of the entity identifiers is clearly not mandatory, hence the impersonation attack should be applicable to the IEEE variant of SPEKE. On the other hand, we cannot state the same for the ISO/IEC variant of SPEKE, because its definition of the shared secret is not sufficiently clear. Neither standard provides any clear formula about the definition of the shared secret. This is unsatisfactory, especially because the detail here has critical security implications.

For a more concrete discussion, let us denote Alice's identifier as $\hat{A}$, Bob's identifier as $\hat{B}$ and the session identifier as SID. One straightforward way to include all these identifiers is: $s = H(\text{Password}\|\hat{A}\|\hat{B}\|\text{SID})$. But this implies a preferred order of the parties' identifiers, which need to be agreed beforehand. A slightly better definition is as follow: $s = H(\text{Password}\|\min(\hat{A},\hat{B})\|\max(\hat{A},\hat{B})\|\text{SID})$. Yet it remains questionable how the SID should be defined and by whom. In the general case, the unique session ID is decided by both parties as part of the key exchange process, but this usually requires extra rounds of communication. The requirement for extra rounds is undesirable as it would remove the most notable advantage of SPEKE in terms of its optimal one-round efficiency. The way that the two standards address this extra-round issue is by defining the session ID (together with the entity identifiers) as part of the "prior shared parameters" before the key exchange. Hence, the SPEKE protocol remains one-round.

However, we believe the above solution in the standards is inappropriate, as it does not address the real problem. It is not a safe assumption that the user must know the other party's identifier or a session identifier before any communication is started. It is often difficult enough for a user to remember her own user name (identifier) and password; requiring the user to remember the other entity's (exact) identifier will only add to the burden on the user's memory and consequently make a PAKE protocol less useful. When the PAKE session fails, it will be no longer clear if that is due to the mismatch of the password, or simply because the user misremembered the identifiers. The user identifiers, as well as the session ID, should be determined as part of the key exchange process. In Section 5.3, we will present a solution that addresses the identified attacks without requiring any extra memory burden, in the meanwhile still keeping the SPEKE protocol one-round only.

Effect on Key-malleability attack The inclusion of identifiers for one or more entities and the specific session into the definition of the password-based string has no effect in preventing the key-malleability attack.

Alice ($\hat{A}$)		Bob ($\hat{B}$)
Select $x \in_R [1, q-1]$. Compute $M = g^x$	$\xrightarrow{\hat{A}, M = g^x}$	Check $M \in [2, p-2]$
Check $N \in [2, p-2]$	$\xleftarrow{\hat{B}, N = g^y}$	Select $y \in [1, q-1]$. Compute $N = g^y$
Alice Computes: $\kappa_a = H\left(\min(\hat{A}, \hat{B}), \max(\hat{A}, \hat{B}), \min(M, N), \max(M, N), N^x\right)$		
Bob Computes: $\kappa_b = H\left(\min(\hat{A}, \hat{B}), \max(\hat{A}, \hat{B}), \min(M, N), \max(M, N), M^y\right)$		

Fig. 3. Patched SPEKE

5.3 Countermeasures and suggested changes to standards

There are several reasons to explain the cause of the two attacks. First, there is no reliable method in SPEKE to prevent a sent message being relayed back to the sender. Second, there is no mechanism in the protocol to verify the integrity of the message, i.e., whether they have been altered during the transit. Third, no user identifiers are included in the key exchange process. It may be argued that all these issues can be addressed by using a Zero Knowledge Proof (ZKP) (as done in [4]). However, in SPEKE, the generator is a secret, which makes it incompatible with any existing ZKP construction. Since the use of ZKP is impossible in SPEKE, we need to address the attacks in a different way.

Our proposed solution is to redefine the session key computation. Assume Alice sends $M = g^x$ and Bob sends $N = g^y$. The session key computation is defined as follows:

$$\kappa = H\left(\min(\hat{A}, \hat{B}), \max(\hat{A}, \hat{B}), \min(M, N), \max(M, N), g^{xy}\right) \tag{1}$$

The patched SPEKE protocol is summarized in Fig. 3. When the two users are engaged in multiple concurrent sessions, they need to ensure the identifiers are unique between these sessions. As an example, assume Alice and Bob launch several concurrent sessions. They may use "Alice" and "Bob" in the first session. When launching a second concurrent session, they should add an extension to make the identifier unique – for example, they may agree at the protocol level to start the extension from "1" and increment by one if a new concurrent session is created. Thus, the actual user identifiers become "Alice-1" and "Bob-1" in the second session. In the third session, the user identifiers become "Alice-2" and "Bob-2", and so on. As long the user identifiers are unique between concurrent sessions, the use of the extra session identifier does not seem needed.

The new definition of the session-key computation function in Eq. 1 should address the impersonation and key-malleability attacks in Section 4 (and also the "unknown-key share" attack and the generic attack reported by Tang and Mitchell [9]). This is achieved without having to involve explicit key confirmation, so the key confirmation can remain as "optional" as it is in the current standards. Furthermore, this countermeasure preserves the optimal one-round efficiency of the original SPEKE protocol.

There is an alternative solution, which is to make the definition of a shared low-entropy secret more explicit in the standards. One way is to define the shared secret as below:

$$s = H\left(\text{Password}\|\min(\hat{A},\hat{B})\|\max(\hat{A},\hat{B})\right) \tag{2}$$

In the above definition, the session identifier SID is not included, as the concept seems to have been absorbed in the user identifiers as long as they are ensured to be unique between concurrent sessions.

Comparing the two solutions, we recommend the first solution in Eq. 1 (also see Fig. 3) for the following reasons.

- The first solution is more flexible to accommodate pre-computation of g^x and g^y. In the second solution, the user must know the identifier of the other party before the key exchange, which effectively prevents pre-computation.
- The first solution is more round-efficient. Alice and Bob do not have to know the exact identifier of the other party before starting the key exchange. But in the second solution, Alice and Bob may need an extra round before they are able to compute the generator g.
- The first solution is computationally more efficient. Because the generator g is unchanged for the same password, it only needs to be computed once. In comparison, the generator needs to be re-computed with any change in the user identifiers. (This may not make much difference in terms of computation if a safe prime is used, but it can significantly decrease performance in some other group settings.)

A further suggestion we would like to make for both standards is to reconsider the definition of the key confirmation method. The existing method, as defined in ISO/IEC 11770-4 and IEEE 1363.2, breaks the symmetry of the protocol (the key confirmation cannot be completed within one round). The key confirmation method in the original SPEKE paper [7] has the same limitation.

Our rationale for suggesting this change is not based on security considerations, but on the grounds of round efficiency. The key confirmation method defined in the original SPEKE paper [7] and the two standards [10, 11] cannot be completed in one round. We use the method defined in [7] as an example. If both parties attempt to initiate the explicit key confirmation at the same time, i.e., Alice sends $H(H(\kappa))$ and without receiving Alice's message, Bob also sends $H(H(\kappa))$. In that case, they may enter a deadlock and may have to abort the session and restart a new one. The chance of such an occurrence would be non-negligible in a high-latency network.

The solution we propose is based on the key confirmation defined in NIST SP 800-56A Revision 1 [1]. It works as follows:

$$\begin{aligned}\text{Alice} \rightarrow \text{ Bob } &: \text{HMAC}(\kappa, \text{“KC\_1\_U”}\|\hat{A}\|\hat{B}\|g^x\|g^y)\\ \text{Bob } \rightarrow \text{Alice} &: \text{HMAC}(\kappa, \text{“KC\_1\_U”}, \|\hat{B}\|\hat{A}\|g^y\|g^x)\end{aligned}$$

In the above key confirmation method, HMAC is a hash-based MAC algorithm and the string “KC_1_U” refers to unilateral key confirmation [1]. There

SPEKE protocol variants	Round efficiency	Impersonation attack	Key-malleability attack
Original SPEKE with KC	3	Yes	Yes
Original SPEKE without KC	1	Yes	Yes
SPEKE in IEEE P1363.2 with KC	3	Yes	No
SPEKE in IEEE P1363.2 without KC	1	Yes	Yes
SPEKE in ISO/IEC 11770-4 with KC	≥ 3	Maybe	No
SPEKE in ISO/IEC 11770-4 without KC	≥ 1	Maybe	Yes
SPEKE in IETF I-D with KC	3	Yes	No
SPEKE in IETF I-D without KC	1	Yes	Yes
Patched SPEKE with KC	2	*No*	*No*
Patched SPEKE without KC	1	*No*	*No*

Table 1. Summary of results

is no dependence between the two flows, so Alice and Bob can send messages in one round.

5.4 Summary of results

We summarize the applicability of the reported attacks on various variants of SPEKE in Table 1. For the completeness of discussion, we also include the version of the SPEKE protocol defined in the IETF Internet Draft[4], authored by the original SPEKE designer Davlid Jablon. In this Internet Draft, the entity identifiers are not included into the default definition of the shared secret. From the Draft, "*... in a peer-to-peer application using SPEKE, both parties may compute <g> directly from the shared password.*". The key confirmation function defined in this Internet Draft is basically the same as that in IEEE P1363.2.

With the exception of ISO/IEC 11770-4, all previous versions of SPEKE are vulnerable to the impersonation attack regardless of whether the key confirmation is in place. In ISO/IEC 11770-4 [11], we cannot determine the applicability of the same attack because the wording in the standard is not sufficiently clear. Hence, we mark "Maybe" instead of "Yes" in the table.

The key-malleability attack is applicable to the original SPEKE (patched against the exponential-equivalence attack [8]) regardless of whether the key confirmation is used. In both the IEEE and ISO/IEC standards (and also in the submitted IETF Internet Draft), the key confirmation is modified to include the key exchange messages. Hence, the key-malleability attack no longer works when the modified key confirmation is used. But the key confirmation is marked as "optional" in these standards. Therefore, the versions that do not use explicit key confirmation are still vulnerable to the key-malleability attack.

In this paper, we propose two changes to the standardized SPEKE protocol: one is to redefine the session key computation based on Equation 1 and the

[4] The latest draft version is "02", dated 22 October, 2003. See `http://www.ietf.org/archive/id/draft-jablon-speke-02.txt`

other one is to redefine the key confirmation function based on NIST SP 800-56A Revision 1 [1]. The first change addresses both the impersonation and the key-malleability attacks. The second change allows the key confirmation to be completed in one round.

Our patched SPEKE preserves the overall round-efficiency in the optimal manner. By comparison, in the original SPEKE paper, the IEEE 1363.2 standard and the IEFT Internet Draft, the specified protocol is one-round without explicit key confirmation, and is three-round with explicit key confirmation. In ISO/IEC 11770-4, it is not clear if the entity/session identifiers must be included into the definition of the shared secret. If such an inclusion is mandatory, it would generally need an extra round to send the entity/session identifiers before the key exchange.

Finally, we examine how the SPEKE protocol is actually implemented in practice, particularly with regard to whether or not the entity/session identifiers are included. In practice, SPEKE has been used by Blackberry for secure messaging. In this implementation, only the password (no entity/session identities) is used to derive the generator of the designated group. From the on-line documentation about the derivation of the generator[5]: "*The function applies the ECREDP-1 primitive to the password to derive a generator point.*" Hence, the impersonation attack is in principle applicable to the protocol that underpins the Blackberry application. This does not necessarily mean the Blackberry application must be insecure, since it also depends on the context of an application and other implementation details (e.g., if the application supports parallel sessions). We leave this as a subject for further investigation.

6 Conclusion

In this paper, we present two new attacks on SPEKE, a protocol that has been included in the IEEE P1363.2 and ISO/IEC 11770-4 standards, and deployed in commercial products. The first attack indicates a practical flaw that needs to be addressed, while the second attack has an unfavourable theoretical implication. We explain the differences between the original SPEKE protocol and its variants defined in both standards and show how these differences are critically relevant to the presented attacks. We suggest concrete changes to both standards to address the issues identified in this paper.

Acknowledgement

We thank the anonymous reviewers from SSR'14, Brian Randell, Liqun Chen and Chris Mitchell for many useful comments.

[5] `http://developer.blackberry.com/native/reference/core/com.qnx.doc.crypto.lib_ref/topic/hu_ECSPEKEKeyGen.html`

References

1. E. Barker, D. Johnson, and M. Smid, "Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography (revised)", NIST Special Publication 800-56A, March 2007. Available at `http://csrc.nist.gov/publications/nistpubs/800-56A/SP800-56A_Revision1_Mar08-2007.pdf`
2. S. Bellovin and M. Merritt, "Encrypted Key Exchange: password-based protocols secure against dictionary attacks," Proceedings of the IEEE Symposium on Research in Security and Privacy, May 1992.
3. F. Hao, "On robust key agreement based on public key authentication", Proceedings of the 14th International Conference on Financial Cryptography and Data Security (FC'10), Tenerife, Spain, LNCS 6052, pp. 383-390, 2010.
4. F. Hao, P. Ryan, "Password authenticated key exchange by juggling," Proceedings of the 16th Workshop on Security Protocols (SPW'08), Cambridge, UK, LNCS 6615, pp. 159-171, 2008.
5. "Entrust TruePass Product Portfolio: Strong Authentication, Digital Signatures and end-to-end encryption for the Web Portal," Technical Overview, Entrust Inc., July 2003. Available online at `http://www.entrust.com/wp-content/uploads/2013/05/entrust_truepass_tech_overview.pdf`
6. "BlackBerry Bridge App and BlackBerry PlayBook Tablet," Security Technical Overview - Version 2.0, Research In Motion Ltd., February 2012. Available online through Blackberry Knowledge Base at `http://btsc.webapps.blackberry.com/btsc/microsites/searchEntry.do`
7. D. Jablon, "Strong password-only authenticated key exchange," *ACM Computer Communications Review*, Vol. 26, No. 5, pp. 5-26, October 1996.
8. M. Zhang, "Analysis of the SPEKE password-authenticated key exchange protocol," *IEEE Communications Letters*, Vol. 8, No. 1, pp. 63-65, January 2004.
9. Q. Tang, C. Mitchell, "On the security of some password-based key agreement schemes," International conference on Computational Intelligence and Security (CIS), LNCS Vol. 3802, pp. 149-154, 2005.
10. IEEE P1363 Working Group, P1363.2: Standard Specifications for Password-Based Public-Key Cryptographic Techniques. Draft available at: `http://grouper.ieee.org/groups/1363/`
11. International Standard on Information Technology, Security Techniques, Key Management, Part 4: "Mechanisms based on week secrets," ISO/IEC 11770-4:2006.